AI Assisted Coding

Lab-7.3

2303A51383

Batch – 06

Lab 7: Error Debugging with AI – Systematic Approaches to Finding and Fixing Bugs

---

Lab Objectives

* To identify and fix syntax, logic, and runtime errors using AI tools.
* To understand common programming bugs and AI-assisted debugging.
* To evaluate AI explanations for different types of errors.
* To build confidence in structured debugging practices.

Lab Outcomes

After completing this lab, students will be able to:

* Detect and correct syntax, logic, and runtime errors.
* Interpret AI-suggested debugging fixes.
* Apply systematic debugging strategies.
* Refactor faulty Python code correctly.

Task 1: Fixing Syntax Errors

Scenario

A Python function contains a syntax error due to a missing colon.

Buggy Code (Input to AI Tool)

```python
def add(a, b)
   return a + b
```

AI-Identified Issue

* Python function definitions must end with a colon (`:`).

* Missing colon causes a **SyntaxError**.

Corrected Code

```python
def add(a, b):
    return a + b

print(add(5, 3))
```

Output

```
8
```

AI Explanation & Analysis

* The colon tells Python where the function block starts.

* Without it, Python cannot parse the function.

* AI correctly detected and fixed the syntax error.

Task 2: Debugging Logic Errors in Loops

Scenario

A loop runs infinitely due to incorrect increment logic.

Buggy Code

```python
i = 1
while i <= 5:
    print(i)
```

AI-Identified Issue

* Variable `i` is never incremented.

* Condition `i <= 5` always remains true.

* Causes an **infinite loop**.

Corrected Code

```python
i = 1
while i <= 5:
    print(i)
    i += 1
```

Output

```
1
2
3
4
5
```

AI Explanation & Analysis

* Loops must move toward a terminating condition.
* Incrementing `i` ensures loop termination.
* AI correctly fixed the logical error.

Task 3: Handling Runtime Errors (Division by Zero)

Scenario

A program crashes due to division by zero.

Buggy Code

```python
def divide(a, b):
    return a / b

print(divide(10, 0))
```

AI-Identified Issue

* Division by zero raises a **ZeroDivisionError**.
* Program crashes at runtime.

Corrected Code Using try-except

```python
def divide(a, b):
    try:
        return a / b
    except ZeroDivisionError:
        return "Error: Division by zero is not allowed"

print(divide(10, 0))
```

Output

```
Error: Division by zero is not allowed
```

AI Explanation & Analysis

* `try-except` prevents program crashes.
* AI added proper exception handling.
* Makes the function safer and more reliable.

Task 4: Debugging Class Definition Errors

Scenario

A class constructor is incorrectly defined without `self`.

Buggy Code

```python
```

```
class Student:
    def __init__(name, roll):
        name = name
        roll = roll
```

AI-Identified Issue

* `self` is missing in the constructor.
* Instance variables are not properly assigned.

Corrected Code

```python
class Student:
    def __init__(self, name, roll):
        self.name = name
        self.roll = roll

    def display(self):
        print(self.name, self.roll)
```

s1 = Student("Preetham", 101)

s1.display()

```

---

### **Output**

```

Preetham 101

```

---

### **AI Explanation & Analysis**

* `self` refers to the current object.

* Without `self`, instance variables cannot be stored.

* AI correctly fixed object-oriented structure.

---

## **Task 5: Resolving Index Errors in Lists**

### **Scenario**
```

A program crashes due to accessing an invalid list index.

Buggy Code

```python
numbers = [10, 20, 30]
print(numbers[5])
```

AI-Identified Issue

* Index 5 does not exist.
* Causes an **IndexError**.

Corrected Code Using Bounds Check

```python
numbers = [10, 20, 30]

index = 2
```

```
if index < len(numbers):
    print(numbers[index])
else:
    print("Index out of range")
```

Alternative Solution Using try-except

```python
try:
    print(numbers[5])
except IndexError:
    print("Index out of range")
```

Output

```
Index out of range
```

AI Explanation & Analysis

* AI suggested safe list access methods.
* Prevents program crashes.
* Improves robustness of the code.

Overall Conclusion

* AI tools effectively detect **syntax**, **logic**, **runtime**, **OOP**, and **indexing** errors.
* AI-generated fixes are accurate and beginner-friendly.
* Human review is essential to understand and validate AI suggestions.
